# FireGrid: An e-infrastructure for next-generation emergency response support

Han, L., Potter, S., Beckett, G., Pringle, G., Welch, S., Koo, S-H., Wickler, G., Usmani, A., Torero-Cullen, J., & Tate, A. (2010). FireGrid: An e-infrastructure for next-generation emergency response support. Journal of Parallel and Distributed Computing, 70(11), 1128-1141. 10.1016/j.jpdc.2010.06.005

# FireGrid: An e-Infrastructure for Next-Generation Emergency Response Support

**Liangxiu Han[a *], Stephen Potter[b], George Beckett[c], Gavin Pringle[c], Stephen Welch[d], Sung-Han Koo[d], Gerhard Wickler[b], Asif Usmani[d], José L. Torero[d] and Austin Tate[b]**

**[a]School of Informatics, The University of Edinburgh, Edinburgh, UK**

**[b]Artificial Intelligence Applications Institute, The University of Edinburgh, Edinburgh, UK**

**[c]Edinburgh Parallel Computing Centre, The University of Edinburgh, Edinburgh, UK**

**[d]BRE Centre for Fire Safety Engineering, The University of Edinburgh, Edinburgh, UK**

[*] Corresponding Author: liangxiu.han@ed.ac.uk

**Abstract**

The FireGrid project aims to harness the potential of advanced forms of computation to support the response to large-scale emergencies (with an initial focus on the response to fires in the built environment). Computational models of physical phenomena are developed, and then deployed and computed on High Performance Computing resources to infer incident conditions by assimilating live sensor data from an emergency in real time – or, in the case of predictive models, faster-than-real time. The results of these models are then interpreted by a knowledge-based reasoning scheme to provide decision-support information in appropriate terms for the emergency responder. These models are accessed over a Grid from an agent-based system, of which the human responders form an integral part. This paper proposes a novel FireGrid architecture, and describes the rationale behind this architecture and the research results of its application to a large-scale fire experiment.

**Keywords:** Emergency Response, Grid, High Performance Computing, Multi-Agent System, Knowledge-Based Reasoning, Fire Simulation Model

## 1. Introduction

To minimise losses to life and property during emergencies, decisions have to be made by people in a timely manner; and the availability of relevant information is critical if the right decisions are to be made. Advances in Information Technology (IT) provide alternative channels for information flow, with the potential to place additional crucial information at the disposal of decision makers during an emergency response. The vision for the FireGrid project [3,5] is of a generic software architecture that provides an integrated IT solution primarily for supporting the response to emergency events. In the first instance, the project has focused on the emergency response to fires in the built environment.

For obvious reasons, fire-fighters will rarely be aware of the actual conditions within a building during a fire incident. Consequently, they will be compelled to make intervention decisions based on the limited information provided by their senses, and drawing heavily on their training and on their past experiences of fires. However, given the complex nature of fire, for even the most experienced of fire-fighters the interpretation and extrapolation of conditions is a difficult task. Advances in several technologies when taken together suggest a new approach to the problem of assessing conditions during fire emergencies:

- Developments in sensor technology, and a reduction in unit cost, offer the prospect of deploying large-scale, robust and cost-effective sensor networks within buildings;
- Advances in the understanding of fire and related phenomena have resulted in sophisticated computational models which might be used to interpret sensor data;
- The availability of Grid infrastructure for distributed High Performance Computing (HPC) and data processing suggests a platform on which these models could be run in real and faster-than-real time, making their use in emergencies a practical proposition.

The FireGrid concept aims to improve – both in range and quality – the information available to fire-fighters. This requires capturing information in real time, interpreting information accurately, and presenting information in an accessible and concise manner.

In this paper, we propose a novel system architecture in which real-time sensor data is captured, filtered and stored; in which computational models are developed, deployed on High Performance Computing resources and, in the event of an emergency, run using ‘live’ sensor data in real-time to steer the model output; and in which the results from these models are interpreted using knowledge-based reasoning operating within an agent-based command-and-control layer. A Grid middleware component provides a uniform interface for interactions among the

agent-based command-and-control layer, computational models and HPC resources, and Artificial Intelligence (AI) techniques are used to interpret the outputs of the models for the end user. The prototype of a FireGrid system incorporating all of these aspects has been demonstrated and evaluated through its application in the context of a large-scale fire experiment.

The rest of this paper is structured as follows. Section 2 presents some related work and challenges on emergency response and management systems. Section 3 describes the rationale behind the FireGrid architecture; Section 4 describes the components of developed architecture; Section 5 presents an experimental evaluation of the FireGrid architecture through its application to a large-scale fire demonstration; and Section 6 provides some concluding remarks.

## 2. Emergency Response and Management Systems

In recent years, a significant amount of effort has been devoted to providing computer support for emergency management. Geographical Information System-based applications have been developed to help decision makers to analyze, manage and respond to emergency situations by situating incident information in its geospatial context [10,24]; specific examples include emergency management systems for containing chemical and nuclear pollutants [12], for monitoring the risk of oil pollution [25], and for tracking and visualising the predicted course of hurricanes [26]. An Artificial Intelligence-based emergency response system [4] has been applied to environmental monitoring, and an example of the use of a knowledge-based model for decision support during flood emergencies can be found in [13]. Multi-agent systems have been used in complicated and large environmental emergency systems [15]; WIPER [22], designed as a multi-agent system, is an attempt to provide emergency planners and responders with an integration of web services and service oriented architecture technologies. DrillSim [19] is an augmented reality multi-agent simulation environment for testing IT solutions in emergency contexts. DEFACTO [23] incorporates state-of-the-art artificial intelligence, 3D visualization and human interaction reasoning into a system for responder training.

Notwithstanding these successes, FireGrid introduces a further significant and overriding concern, namely the collection, management and use of dynamic information, which is so critical to emergency response. Most of the existing work provides information that is pre-compiled and prepared beforehand, rather than being generated in real time. While this can be appropriate for certain types of incident, the provision of up-to-date information in the event of an unforeseen highly dynamic emergency presents a set of challenges all its own, especially as the scale of the incident grows and information from dispersed and diverse sources must be integrated before it can be used to make

decisions. This dynamism has implications too for the human interfaces to the system, as the pressures of the situation, the rate of information flow and poor visualizations can conspire to overwhelm the decision maker.

To address these challenges, a FireGrid system should, firstly, provide status information about resources, incidents or events in real time; secondly, provide data management functionality that dynamically locates and transports datasets between storage systems and applications; and finally, be granted sufficient autonomy to facilitate the responder's task by performing filtering and interpretation of information prior to presentation.

To meet these goals we propose a novel FireGrid architecture. In the following sections, we present the rationale behind the FireGrid architecture, describe technologies used in the system architecture, and demonstrate the research results of its application to a large-scale fire experiments.

## 3. The Rationale Behind FireGrid: From Data To Decision

In essence, the objective of a FireGrid system is dynamic data manipulation: we envisage a system that acquires sensor data from the environment of a fire, that interprets and analyses this data to produce relevant information, and that relays this – presented in an appropriate form – to the response decision-makers. Hence, at an abstract level, an architecture for a FireGrid system must encompass data gathering, data interpretation and information presentation.

### 3.1 Data Gathering

For a fire incident, the data gathering process consists of the continual collection of data from the environment of the fire, and the transfer and storage of this data. Different sensors such as smoke detectors, thermocouples (temperature sensors), and CO and $CO_2$ detectors are deployed in the building for continuous monitoring of its internal state. In the event of fire, the output from each of these sensors provides potentially valuable data for input to computational models. Typically, these sensors will be polled in batch mode periodically by one or more 'data loggers', physical devices with which groups of sensors have some direct communications link, and which also convert the signals received from the sensors into their corresponding quantities (so, for instance, voltage levels reported by thermocouples are converted into the corresponding temperature readings). In modern systems, these steps are automatic, and at this point these data values can be accessed and stored in a database. Since sensors (or their lines of communication) can be noisy or can fail because of manufacturing flaws or the extremes of the fire incident itself, the data first needs to be verified and filtered to provide some measure of the accuracy and quality of data received from the building. To address this issue, we have developed a constraint-based filtering algorithm for validating sensor readings [27]. The data values must also be tagged with meta-data describing their origin and sampling time, before being transferred in real-time into some data storage facility. While a higher sampling rate produces more

data and increases the potential for downstream data processing, it also means more data must be processed, transferred and stored in the same amount of time. Hence, some practical compromise must be found.

For a deployment in a particular building, a central database is created for storing all information about and generated by the use of a FireGrid system. In addition to the *dynamic* real-time data that is being continuously fed from the sensor network, the database also holds *static* data, that is (reasonably) time-invariant data such as the geometry/layout of the building, the types of sensors and their locations, the types and material of furniture and the location of fire suppression systems. Apart from physical and material properties, static data may also include pre-computed scenarios of fire development, which can be compared with the real fire state so as to steer computational models.

### 3.2 Data Interpretation

The data interpretation process is one of transforming the collected data into descriptions of the current status and predictions of the development of the incident; clearly, these interpretations should produce results that are relevant to decision-making during an emergency response. In the first instance, this transformation is done using computational interpretation and simulation models of varying sophistication and complexity. Such a model might involve little more than a simple calculation over the latest sensor readings to provide, say, the current maximum temperature in a room. Richer modelling approaches that exploit Computational Fluid Dynamics (CFD), Finite Element (FE) methods and other techniques are potentially powerful tools for representing aspects of fire development and associated phenomena such as structural integrity, smoke movement and human egress. However even these prove inadequate for the representation of certain phenomena, where coupled methodologies are required to provide linkage between different aspects of real problems. Whilst some of these exist, for example in coupled CFD-FE treatments such as that available in ANSYS CFX [2] and coupled fire-egress models such as CRISP [8,9], many are highly demanding computationally or remain in the research domain. For the purposes of FireGrid, where it is intended to use these models to interpret and predict in real time, ideally a model would be independent of any particular context (such as a specific building) or incident, and would rely for its initial conditions on information acquired from the system database, with updates based on later sensor data whenever appropriate. In this context, the computational cost of running a simulation arises from the interplay of a number of factors: the complexity of the underlying model; its scope in terms of the size of the physical and temporal space which it considers; the amount of data to be processed; and the desired accuracy and precision of the results. At their most costly – say, models that attempt to extrapolate from the data and produce accurate, precise and far-reaching predictions of the course of the incident – these simulations are computationally intensive, and require a proportional amount of computational

power if results are to be produced in a timely fashion. Furthermore, strategies for effective and efficient data communication are required to support this processing. These considerations have led to the adoption of HPC, with appropriate simulation models deployed on specific HPC resources (and optimised to run most effectively on those resources), and invoked in the data context provided by a Grid Computing infrastructure. We shall have more to say about the use of HPC and the Grid below in sections 4.2 and 4.3.

We have developed a computational model entitled "K-CRISP" [16,17], an extension of CRISP, for simulating the fire in the experiment described in Section 5. This model is a sensor-linked extended zone model for fire development and simplified structural response for multiple rooms coupled with a model of human behaviour. It can be run in a Monte-Carlo fashion for risk analysis, with multiple alternative scenarios for the development of the fire after ignition time generated from a set of initial conditions [8]. The model predicts what the sensor measurements will be for given input conditions (for example, fire location, item properties, and doors open and closed) and compares its initial predictions to the real data collected from sensors. Using a procedure based (in part) on Bayesian inference, the model adjusts its parametric space from which new scenarios are generated, thereby identifying fire scenarios which are progressively better matches to the sensor readings. This data-driven approach for 'steering' the model has been shown to be effective in directing the evolution of the model parameters, and hence of the predictions it makes. The model is able to provide some indication, particularly in the short term, of the predicted evolution of the fire, human egress behaviour and structural integrity (that is, the likelihood of collapse). Moreover, by virtue of the fact that information on probabilities is an integral part of the simulation output, information can be derived to provide end users with an indication of the likelihood of various hazard scenarios. As might be expected, the best forecasts are those for the immediate future and for relatively simple fires, with progressively less confidence at longer lead times and in more complex scenarios. Nevertheless, given the uncertainties in real fire development, the benefits of more detailed model representations may be marginal and the system developed thus far is considered to be an appropriate engineering approach to the problem, providing information of potential benefit to emergency response. The Monte-Carlo technique underlying this model readily lends itself to parallelisation and thus to maximum exploitation of however many processing units are available at the HPC resource.

However, while it may be potentially useful for emergency response, unless the model has been developed specifically for this purpose its output is unlikely to be in a form that is immediately comprehensible to the emergency responder: it will likely need further interpretation to extract the relevant information. And when faced with the possibility of information from a number of different models, to be useful any system would need to be able to reconcile these into a coherent whole for the responder. Moreover, we would like a system in which the

communication occurs in two directions, with responders able to request specific information of the system (requests which may then lead to the invocation of specific models). The need to achieve this level of interpretation and interaction for emergency response has led us to build a knowledge-based reasoning scheme to interpret the output of computational models. This scheme incorporates query answering, belief revision and hazard inference, mechanisms which are underpinned by the FireGrid system ontology.

### 3.2.1 The Knowledge-Based Reasoning Scheme: The FireGrid Ontology

As envisaged, a FireGrid system would allow its fire-fighter user to relate the output from simulation and interpretation models to the risks faced in the current incident. In order to do this, some common ground must be identified within (or else imposed upon), on the one hand, the information emerging from the computer models, and on the other, that understood by fire-fighters as potentially relevant to the risks they face. In other words, in AI terms it is necessary to establish an *ontology* for use within the system. An ontology sets out in explicit terms the key concepts in the domain, along with the relationships that hold among them, and in so doing it defines the terminology to be used when referring to these concepts. Based on discussions with both fire-fighters and modellers, we were able to identify a number of common concepts understood by both. In its underlying approach this ontology draws on other ontological work, in particular the categories defined in DOLCE [11]. Below we discuss some of the key ontological definitions, namely the distinction between *State-Parameters* and *Events*; the expression of *Hazards*; and the ontological assumptions made about space and time. A high-level view of part of the FireGrid ontology is shown in Fig.1.

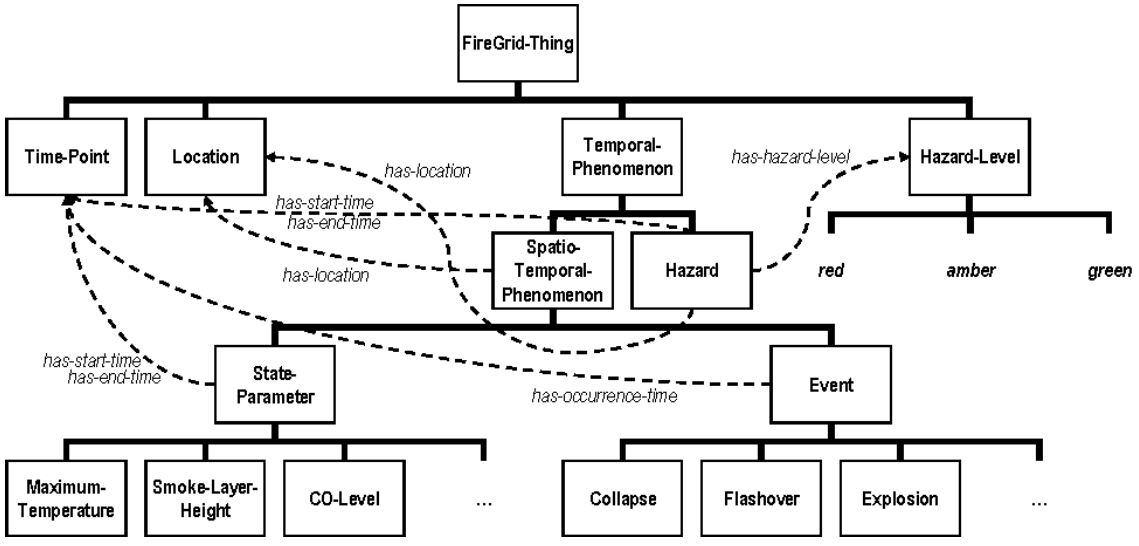


**Fig. 1 A high-level view of part of the FireGrid ontology**

In Fig.1, concepts (e.g., *Time-point*, *Location*, etc.) shown in boxes are arranged in a sub-concept hierarchy below the most generic FireGrid-Thing concept. Dashed lines describe relationships among concepts. So, for

instance, the concept of Event is a sub-concept of Spatio-Temporal-Phenomenon; it in turn has sub-concepts corresponding to different types of event; and it has specific relationships with Time-Point and, via its parent concept, with Location. Terms in the hierarchy shown without boxes (here, red, amber and green) correspond to instances of the parent concept (*Hazard-Level*).

**(1) State Parameters and Events**

The FireGrid ontology makes a distinction between *State-Parameters* and *Events*. State parameters are quantities that are considered to be continuously measurable for some place over some duration of time. Illustrative sub-concepts include *Maximum-Temperature* and *Smoke-Layer-Height*. Events, in contrast, are considered to be instantaneous occurrences at some location, such as *Collapse* or *Explosion*. Furthermore, it is asserted that a certain sub-concept of *Event* can only occur once (if at all) at a particular location during the incident – although in reality multiple occurrences of an *Event* sub-concept are certainly possible, this constraint was imposed to ease the conceptual difficulties of knowing which occurrence of a particular type of *Event* a model is referring to, when, for example, it provides a revised set of predictions. It is intended that both *State-Parameters* and *Events* can be derived (albeit through additional knowledge-based interpretation) from the output of models – and, moreover, if a given model does not produce a recognized *State-Parameter* or *Event*, then its usefulness for and relevance to FireGrid is, at best, questionable.

**(2) Hazards**

From the fire-fighter's perspective, the values of *State-Parameters* and the occurrence of *Events* can be related (again, through knowledge-based interpretation) to the concept of a *Hazard*: a *Hazard* is defined as something that can impinge upon the operational safety of fire-fighters at a particular place for some particular duration. For the purposes of relating this to the simplified 'traffic light' paradigm for information presentation (see below), we can define the concept of *Hazard-Level* as being a relative measure of the severity of a *Hazard* that pertains at some time at some location; and, more specifically, we can identify three specific values of *Hazard-Level* and define these in terms directly related to fire-fighting operations:

- A *green Hazard-Level* should be interpreted as "the system is unaware of any specific hazard to fire-fighters operating under normal safe systems of work at this location at this time";
- An *amber Hazard-Level* as "additional control measures may need to be deployed to manage hazards at this location at this time";
- A *red Hazard-Level* as "this location may be dangerous for fire-fighters at this time".

**(3) Space and Time**

As described above each *Hazard-Level* (as well as each *Hazard*, *State-Parameter* and *Event*) is relevant to a particular physical location, which raises the question of the definition and extent of *Location* within the system ontology. This is not as straightforward as may first appear; in models, differentiated spaces (usually) correspond to volumes of gas bounded by physical partitions (and hence correspond to rooms), but this may vary if the model is of either a large space or at a high resolution. For fire-fighters, on the other hand, the notion of *Location* is situation-dependent and dynamic, depending on (among other things) the nature and scale of the building including its vital access and exit routes, the position of the fire incident and any occupants, the tactical operations that are currently underway and so on. To reconcile these views we have adopted a pragmatic approach, defining contiguous locations each of which corresponds to a room in the building in question, since this is one notion that seems to be mutually understood.

Similarly *Hazards*, *State-Parameters* and *Events* all occur in time, and the fire-fighters' decisions relate to both their understanding of what is currently happening and what is predicted to happen in the future. And, as for location, the handling of time within a FireGrid system is not a simple matter. It is necessary that all information in the system is tagged with absolute timestamps, rather than referring to relative times (and it follows that the clocks of system components that generate or present information are synchronized). We shall return to the representation of time in a FireGrid system in the context of the discussion of belief revision given below.

### 3.2.2 The Knowledge-Based Reasoning Scheme: Using the Ontology

The ontology defined in these terms is used to express and communicate the information that is generated by the models. This usually requires the model to be 'wrapped' by appropriate interpretation code, developed with the assistance of the modeller, that is able to interpret the native output of the model in terms of *State-Parameters* or *Events*.

A further use of the ontology is to provide the basis of a 'query language' that enables the system user – through an appropriate user interface agent – to construct and pose to the system requests for specific information (and hence, the ontology here allows the user to communicate with the system). This involves the use of different types of query, with each type intended to elicit particular instances of *State-Parameter* or *Event*. For example, a *confirm-query* is used to request the current value of a designated *State-Parameter*; while a *predict-when-query* is used to request the time of the (first) occurrence of some type of *Event*:

*confirm-query state-parameter-type:* Maximum-Temperature *location:* room-1 [*start-time:* now]

*predict-when-query event-type:* Collapse *location:* room-2

These requests are handled by an autonomous *query manager* agent able to invoke appropriate models. A model is considered appropriate for answering a query if its information-providing capabilities 'match' the query. These capabilities are also expressed in terms of the ontology; consider the following 'rule' that expresses the capabilities of a 'current maximum temperature' model:

IF query-type=confirm-query AND state-parameter-type=Maximum-Temperature

THEN *this model is capable of answering the query*

ELSE *this model is not capable of answering the query*

Given expressions of model capabilities in this form, the query manager should be able automatically to match queries to models. The results generated by the models are then passed back as messages from the query manager agent to the user interface agent, and hence to the responder. See [21] for more details of this query-answering approach.

In addition to enabling effective communications, the second broad use of an ontology is to allow automated reasoning of some form. From the perspective of the end user, a system architecture of the sort described here entails the effective management and interpretation of potentially large amounts of information, derived from different models at different times (and hence potentially conflicting or contradictory), and referring to physical manifestations of the incident at particular points in time and space. To address this task we use two distinct forms of automated reasoning, applied sequentially and cyclically as new information arrives at the responder's interface: first, this information is managed by a belief-revision reasoning mechanism to maintain a consistent description of the current and predicted values of *State-Parameters* and occurrences of *Events*; second, a rule-based reasoning mechanism interprets this description in terms of the hazards it implies.

### 3.2.3 The Knowledge-Based Reasoning Scheme: FireGrid Belief Revision

The information that is presented to the responder is based on a current belief set maintained by the user-interface agent. A *belief* here is some proposition that is held to be true for some location and over some duration. In addition, every belief must have one or more justifications, indicating the rationale for believing it. The justification will usually be one or more messages (sent via the query manager) from the models in the system: the content of messages provides the basis for beliefs. As more information arrives from the models, a process of belief revision is required to maintain the consistency of the set of beliefs. The belief revision required for FireGrid differs in certain important aspects from conventional AI approaches to belief revision such as [1]. Specifically, whereas common

approaches to belief revision operate at an abstract logical and content-independent level, for FireGrid the revision must take into account application-dependent ontological notions.

When a new message arrives from a model, it will correspond to a proposition about either the value of a *State-Parameter* or the occurrence of an *Event* at some time at some location (and it is implicit that this proposition is 'believed' by the model at the time it was sent; in this treatment, we ignore the degrees of belief implied by probabilistic models). The contents of this message have to be considered in the context of the existing beliefs. This process is best illustrated through the use of an example.

Consider a message (which originated in the computational results of a model denoted by the label *model_1*) sent to the user-interface agent consisting of the following elements:

- *sender*: model_1
- *time received*: 12:54:34
- *contents*: Maximum-Temperature = 230°C *in location*: room-A *at time*: 12:54:32

If the agent currently believed nothing about the *Maximum-Temperature* in *room-A* and received the above message, and assuming that the source of the message (that is the *model_1*) is trusted, this message would provide sufficient justification for the agent now believing the contents of the message. Moreover, since nothing else is known about the values of this state parameter in this location, the reasoning would assign a duration to this belief stretching from the current time to some indefinite time in the future. That is, since it is not believed otherwise, an assumption of the belief revision mechanism is that the values of state parameters persist, and hence in this case the maximum temperature would now be believed to remain at 230°C indefinitely – or at least until such time as some other message causes this belief to be revised with a definitive end-point.

If, on the other hand, something is already believed either about the current or the future values of the maximum temperature at this location, then a more complex train of reasoning begins, which attempts to reconcile this message with the existing belief(s). This may involve adjusting durations of beliefs or, where there seems to be a direct contradiction, choosing to adopt one or other of the possibilities and disregarding the other. While this might be done on the basis of, say, the relative trustworthiness of the originating models, in practice we tend to trust models equally, and rely instead on two general principles encoded in the revision mechanism: one of favouring more recent information (and any beliefs it justifies) as being more likely to be true; and a second of favouring interpretations of current state based on sensor data over predictions.

Contradictions become apparent when trying to reconcile inconsistent state descriptions about the same location at the same time. Since we have effectively compartmentalized the incident into distinct locations, determining whether the contents refer to the same location is straightforward. However, determining if the contents refer to the

same time is more problematic; since absolute timestamps are used, the contents of a message and an existing belief about a state parameter may have widely differing values at times that differ by perhaps only fractions of a second. Of course, such a transition in values is possible (often coinciding with some event), but in practice seems more likely to result from the divergence between new information, derived from later sensor readings or more complete simulations, and a belief based on obsolete information. Accordingly, we choose to try to 'smooth' these transitions by defining that if the difference between the start times of a belief and the contents of a message is within a tolerance (set experimentally to 30 seconds) then they refer to the 'same' time. Furthermore, this tolerance helps to overcome the problem that interpretations of 'current' state based on sensor data will always refer to the past due to the inevitable lags and delays in the system; with this tolerance, these interpretations can be assumed to be about 'now' (as would be the case with the example message given above).

A further complexity arises when the content of a message is a prediction – that is, it purports to describe the value of a *State-Parameter* or the occurrence of some *Event* at some location at some future time. While this might be adopted as a belief with a duration as before, the inexorable flow of time will mean that, assuming this belief has not been retracted or modified in the meantime, at some time the prediction will come to refer to the current time, and in the absence of other information a choice must be made about whether or not to accept the predicted value as an actual current value. While reasoning of this sort is difficult to justify on grounds of logical soundness, it can be justified on the basis of a cautious approach to the safety of fire-fighters in the absence of information to the contrary.

### 3.2.4 The Knowledge-Based Reasoning Scheme: Hazard Rule-Based Interpretation

Assuming that the set of beliefs has been revised and is consistent, the next step is to interpret these beliefs by applying a set of *hazard rules* to them. These rules represent expert knowledge about fire-fighting capabilities and practice; an example might be:

| |
|---|
| IF *Maximum-Temperature* ≥ 100°C at location *l* from time $t_1$ until time $t_2$<br>THEN there exists a *Hazard* with *Hazard-Level* = *amber* at location *l* from time $t_1$ until time $t_2$ |

A hazard rule consists of one or more conditions and a single conclusion, which corresponds to an interpretation of the conditions in terms of an instance of a *Hazard* (with associated *Hazard-Level* value) for the time and place in question. In addition, a hazard rule – especially one that refers to less commonly encountered hazards – may have an associated explanation and recommendations. So, for instance, a rule referring to excessive CO levels may offer the explanation that CO levels in that range can "cause headache, fatigue and nausea" alongside the recommendation to "avoid prolonged exposure or consider the use of breathing apparatus".

For each rule, then, a search is made in the set of beliefs for subsets that both satisfy all the conditions and are contemporaneous (that is, which have overlapping durations). If such a subset exists, then the conclusion of the rule can be drawn. An inferred instance of a *Hazard* results in what is essentially a new belief (or in the modification of an existing belief about a *Hazard* with an additional justification), with a duration delimited by the latest start time and earliest end time among the subset of beliefs satisfying the conditions. Note that changes to the belief set can effectively mean that earlier inferences about hazards no longer hold: in AI terms this is a *truth maintenance* problem. However, rather than implement a full-blown Truth Maintenance System, we have adopted the simpler, but less efficient, expedient of re-computing the hazards following changes to the belief set.

Finally, since the application of the rules may have resulted in the inference of multiple simultaneous hazards, to provide a summary of this information that is more readily assimilated by the responder, the inference engine collates these into a single cumulative *Hazard* (and accompanying *Hazard-Level* value) for each location at every time. This is a (relatively) straightforward matter of determining the 'worst' hazard (level) that is believed to apply. So, for instance, if from the state of room-A at the current time, two 'amber' hazards and one 'red' hazard have been inferred then the current overall hazard in room-A will have a hazard level of 'red'. Note that the set of hazard rules is intended to be derived with the assistance of fire-fighting experts; and that, moreover, different rules might apply in different contexts (such as when there are hazards specific to the building in question), allowing the FireGrid system to be tailored accordingly (and hence these rules can be seen equally to form part of the information presentation phase).

### 3.3 Information Presentation

The final stage is information presentation. This process is required to deliver and present the interpretation results to the emergency responder in the most appropriate manner in order to make a quick and correct response. This requires an understanding of the particular capabilities, roles and tasks of the responder, the medium for presenting the information, and the operational context in which the information is delivered, as well as more general theories of situation awareness and human interface design. The content and complexity of the interpretation is also an issue here, since this can encompass both real and projected values and their variations in space and time and can contain uncertainties and errors. Moreover, if results from more than one model are used, the possibility of conflicting or inconsistent interpretations arises. Hence effective presentation can demand the use of sophisticated filtering, interleaving and preferential strategies, tailored to the specific needs of specific users.

As should be evident, the particular presentation adopted depends on the application in question. For our purposes, based on consultations with serving fire officers, we decided to focus on the task of providing information

at the tactical decision-making level; accordingly the most obvious target user is (using UK fire service terminology) the Incident Commander (IC) (or, more realistically, a senior support officer stationed in a command centre at the scene of the incident and detailed to monitor the FireGrid system and report directly to the IC). The IC [14] is:

> "…responsible for the overall management of the incident and will focus on command and control, deployment of resources, tactical planning, the coordination of sector operations…and the health and safety of crews." pp. 15-16.

Rather than being determined in advance, the range of possible incidents and contributing factors means that the response to any given incident is left to the experience and expertise of the IC in question, except when very specific or rare hazards are involved (such as incidents involving hazardous materials or aircraft). However, one decision is effectively universal when dealing with building fires, regardless of specifics: the decision of whether or not to send fire-fighters into the building. Fire-fighters may be sent into a building (and the IC is said to have adopted an *offensive* tactical mode) if and only if the IC considers that in doing so the chances of saving people (especially) or property outweighs the additional risk to fire-fighters. Otherwise a *defensive* tactical mode – the default – is adopted, whereby the fire-fighters say outside the building until such time as either the fire is extinguished and the incident closed, or else conditions are such that they are now considered to make an offensive mode appropriate.

Whether offensive or defensive tactics are adopted, this decision is subject to continuous review by the IC, through a process known as *dynamic risk assessment*. This process, which of necessity is often done rapidly and with incomplete or uncertain information, represents an attempt to rationalise the factors contributing to the tactical mode decision. This process has been identified as the most appropriate target for information from a FireGrid system: this information (and its modes of presentation) should be such as to contribute to the IC's dynamic risk assessment.

Moreover, the pressures of performing this analytical task on the incident ground are such that seemingly conflicting requirements emerged for information to be presented both in a manner that can be rapidly assimilated into this assessment process, and in a way that provides sufficient detailed rationale to allow its careful consideration. From discussions with senior fire-fighters emerged the idea that these requirements might be reconciled at the interface level through a 'traffic light' display (evident in the description of the knowledge-based reasoning in the previous sections) to give an at-a-glance overview of the current status, with a point-and-click facility for delving into the reasons for the colour of light displayed. In interface terms, then, the level of the cumulative current hazard at a particular location is used directly to colour the corresponding traffic light for that location in the graphical user interface (see Fig. 4). Furthermore, feedback from potential users suggested some indication of future hazard would also be useful, and so a second light was added to display the worst hazard level predicted to occur in the future. Clicking within a location causes a pop-up window to appear in which are given the

hazard rules which fired to produce the inferred hazards. In addition a time-line indicates when any hazards are predicted to occur within a time-frame projected into the future (pragmatically set to 15 minutes for our experiments, but different incidents might demand different time-frames); moving a slider allows the user to explore the nature of these hazards.

### 3.4 Summary

Thus, the generic approach for FireGrid is to use sensor data as input into computational models which simulate fire and associated phenomena, and so provide useful interpretations of the current and imminent state of the incident to help responders to make decisions. In the following section we describe how these generic elements are situated in a computational framework that makes a FireGrid system a viable proposition.

## 4. A Sensor-Steered, Grid-Enabled, Agent-Based Infrastructure for Emergency Support

The rationale behind the FireGrid approach described in the previous section allows us to state that the FireGrid architecture consists of four principal components: a data acquisition and storage component for capturing and storing live sensor data; a simulation component for deploying and running computational models on HPC resources; a knowledge-based command-and-control component to provide decision-support for emergency responders; and a Grid middleware component to provide a uniform interface that connects the simulation component and the agent-based command-and-control component. In this section we describe some of the implementational details of each of these components. Fig.2 shows the interactions of these components.

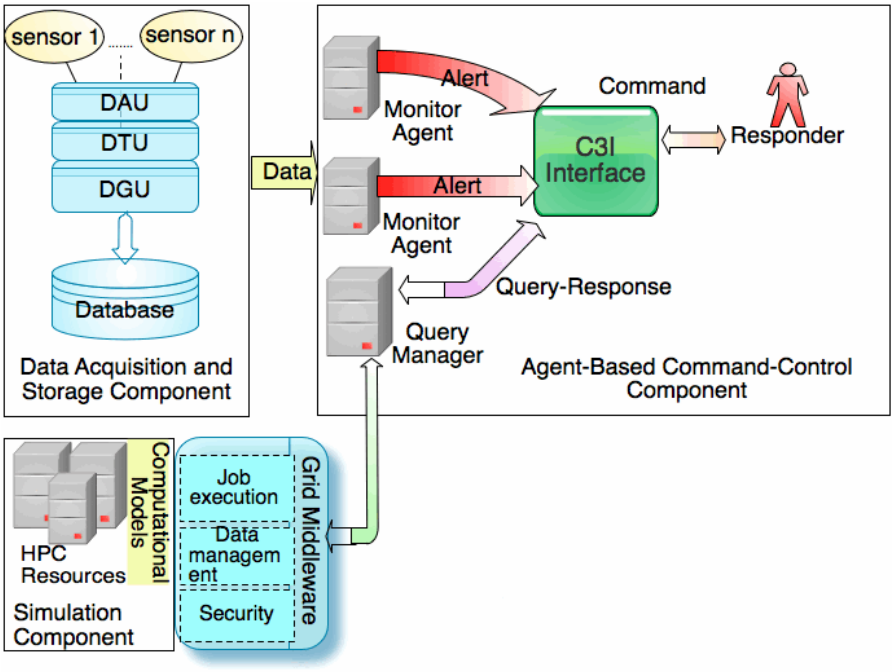


**Fig. 2 The FireGrid architecture for emergency response support.**

### 4.1 Data Acquisition and Storage Component

The role of the data acquisition and storage component is to implement the collection of data as described above in Section 3.1, which introduced the notions of dynamic and static data held in a system database. In practice, collection of dynamic sensor data requires the sequential application of a Data Acquisition Unit (DAU), a Data Translation Unit (DTU), both conventional elements of sensor networks, and a Data Grading Unit (DGU) to capture and validate raw data from various types of sensors. The role of the DAU is to pull raw sensor data (in the form of voltage readings, typically in the range 0V to +/-5V). The role of the DTU is to transform the raw data coming from the data acquisition unit into a form that is appropriate for interpretation (for example, the conversion of thermocouple voltage readings into temperatures). The DTU also time-stamps data (using the enhanced Unix time representation). Time-stamping is vital for the operation of the models and, subsequently, for the knowledge-based reasoning mechanisms that merge the information produced (hence, it is assumed that, multiple DTUs have their internal clocks synchronized). The role of the DGU is to filter the data coming from the data translation unit to attempt to validate its accuracy and reliability before storing it in the database; as a result, every sensor reading has an associated numerical 'quality' rating. For experimentation purposes, a standard relational database has been used to store both the static and the dynamic data, and is hosted on a physically remote server to prevent damage. However, this still represents a single point of failure for the system, and so additional precautions, such as the distribution or even duplication of the database would have to be considered for any actual deployment.

### 4.2 Simulation Component

The simulation component consists of computational models deployed on HPC resources for interpreting and predicting the current status and future development of fire and related phenomena. As described in section 3.2, we have developed a sensor-linked fire simulation model. We intend to exploit the use of HPC resources makes plausible the use of computationally expensive simulations of a fire incident in order to provide information to responders. It is important to realise that the use of HPC does not guarantee faster computation. HPC resources typically provide access to some number of processing units *in parallel*; hence to best exploit the resource, code should be inherently parallel – although parallelizing serial code is rarely a simple task. Furthermore, HPC only provides a speed-up in terms of processing time; code that, for example, requires synchronization or performs multiple file or database accesses may see little or no improvement in run-times. Specific HPC machines will have specific architectures and operating systems, which means that, even where suitable code exists, modifications and recompilations will usually be necessary to port code from one resource to another. A further important point, more specific to FireGrid, is that, since they are valuable resources, access to HPC machines is usually managed through a

scheduling system. Obviously, during emergency response situations it would be unacceptable to have FireGrid jobs held in a queue, so we have an additional requirement, namely that a FireGrid system requires either a dedicated HPC resource of its own, or else (more practically, given the costs involved) it requires access to HPC resources in 'urgent computing mode', enabling it to bypass queues while the resource manager ensures sufficient processing power is available by managing the load on the machine, possibly interrupting running jobs.

K-CRISP, the principal simulation model selected for the experimentation described in the following section, employs a Monte-Carlo technique, involving the generation of large numbers of independent scenarios. As such it is inherently parallel, and so it is well suited to HPC deployment, and an appropriate implementation has been developed. We have used two available HPC resources for experimentation, namely the Edinburgh Compute and Data Facility (ECDF) at the University of Edinburgh and HPCx, the UK National Academic Supercomputer, which was a backup machine in the event of failure, using Linux and IBM AIX operating systems respectively. For ECDF, the 'urgent computing' requirement was fulfilled by isolating a portion of the machine (a single, 8-core computing node) from the main cluster and deploying a separate instance of the batch system (SUN Grid Engine) to manage it as a dedicated resource for FireGrid system jobs. For HPCx, the requirement was fulfilled by the exclusive allocation of computer resource (one 32-core computing node), reserved prior to each experimental run via an advanced reservation request to the general HPCx batch system. The reservation effectively provided a dedicated computing resource for the FireGrid project during each fire experiment.

### 4.3 Grid Middleware Component

The Grid middleware component allows users to access the various computing resources via a uniform computational interface. In general, remote access to HPC resources is often mediated via the Grid, since this provides the following desirable system characteristics [7]:

- Heterogeneity: different hardware and software platforms interacting in a seamless manner.
- Dynamism: placing minimal assumptions on the performance, availability and presence of different components of the system.
- Scalability: the capacity to cope with significant peaks in data generation and movement at key points during use.
- Security: components able to share potentially sensitive data in a secure manner, with trusted third-party systems.
- High performance: providing a level of responsiveness that is in line with the needs of a user with dynamic and evolving requirements.

Functional requirements for the FireGrid system are the provision of a job execution service for invoking models on remote resources, the staging to the remote host of input files for the models, the transfer of output files from the remote host back to the client after job completion, the monitoring of job status, and the provision of security and authorisation services. We have used the Globus Toolkit 4 (GT4) [6] to build the middleware layer, which adopts a client/server model. The CoG Kit [18] has been used for developing a client agent to submit a computing job. The security scheme of the GT4 provides authorisation and authentication to the FireGrid system.

### 4.4 Agent-Based Command-and-Control Component

Users of a FireGrid system need to be able to access the information generated by the models and to formulate requests for specific information (to be fulfilled, where possible, using the outputs of particular models), according to the framework provided by the knowledge-based reasoning scheme described in Section 3. As already seen, this scheme involves the allocation of particular reasoning tasks to specific agents, namely a query manager agent, which interacts with the models to answer requests for information, and a user-interface agent, which manages and interprets the information from the models for its user. This, and the modular approach to construction and deployment that it would provide, led us to adopt an agent-based software model for implementing the command-and-control layer, and relating these knowledge-based reasoning tasks to the other components of the architecture. Hence, in the command-and control layer of the FireGrid architecture, we have two particular types of agent:

- A query manager agent. This agent has the task of attempting to provide specific information that is requested by users (in some cases their interfaces will make these requests autonomously on behalf of users). This is done by interacting with the available interpretation and prediction models, and may involve arranging for resources to be scheduled and managing data for these models through interactions with the Grid middleware layer. The use of the query language described above in Section 3.2.2 allows queries to be formulated and each available model to provide an explicit description of its information-providing capabilities. When a query is received, the query manager will search for any model capable of answering this query. If such a model is found, the query manager will interact with the Grid middleware layer to invoke the model and produce an answer to the query, which it then passes back to the requester. If a capable model is not found, the query manager replies with a message to this effect and awaits the next query.
- One or more Command, Control, Communication and Intelligence (C3I) user interface agents. Each of these provides an interface (with the underlying reasoning mechanisms) conveying the state of the building in question as interpreted by the FireGrid system and allowing the user to interact with the system. The nature of the interface (and, to some extent, its reasoning since the hazard interpretation rules can be role-specific) is

determined by the capabilities and role of its intended user. While, as mentioned in section 3.3, we have concentrated efforts on providing support for the IC, in other contexts a user might be, say, a member of the local security staff responsible for monitoring the building, and instigating evacuation and calling out the emergency services if a fire is detected, but who would not be expected to tackle personally anything but the smallest fires.. These interfaces also allow, where appropriate, users to formulate queries and send them to the query manager.

In addition to these, there may be further agents in the system, used, for instance, to monitor the status of environments (such as a fire alarm agent that monitors sensor data for evidence of a fire and which can alert the user or perform some other action accordingly) or even autonomous or semi-autonomous units able to enact some response to the situation (such as sprinkler systems). In practical terms, agents were developed using the I-X software suite [20], developed by one of the project partners, which provides a generic framework and the underlying technical implementation for providing support for processes occurring among collaborating human and computer agents.

## 5. Experimental Evaluation: A Case-Study

We have prototyped a FireGrid system and tested it in a large-scale fire experiment that was run in the state-of-the-art Burn Hall fire test facility at the Building Research Establishment (BRE), near London, UK. In this section, we describe this experiment.

The experiment involved a fire initiated in a specially constructed rig representing a small 3-room apartment. The notional scenario for the experiment concerned the possibility of occupants trapped in the apartment: the tactical decision was whether or not to send fire-fighters into the building to conduct a search (although no actual fire-fighting activities or any other intervention in the course of the fire was performed during the experiment). A member of the FireGrid team played the role of support officer to the IC (a senior fire officer was among the audience for the experiment).

The primary objectives for this experiment were as follows:

- Integrating simulation models running on HPC resources with live data from sensors in the rig;
- Integrating the simulation models with the agent-based command-and-control layer;
- Demonstrating Grid-enabled, sensor-steered and coupled ensemble HPC simulations;
- Integrating loosely coupled simulation codes and semi-analytic models for extrapolation in order to predict the evolution of the fire and some critical points such as ‘flashover’, egress failures and structural collapse.

## 5.1 Experimental Configuration

The experimental rig consisted of three rooms connected in a T-shape plan by a corridor, as shown in Fig. 3. Each room was a cube of side 2.4m. The rooms were connected by a corridor 3.6m in length and 1.2m wide. Room 1 was to be the location of the source of the fire. It would also contain typical household furniture (sofa, table, television and bookshelves, all potential fuel for a fire). A technician would start the fire by igniting the sofa. A total of 125 sensors placed throughout the rig measured temperatures, heat flux, gas ($O_2$, CO, $CO_2$) concentrations and deformation of structural elements. Values from each of these sensors were polled in batch mode at roughly 3-second intervals, and fed to a database server housed off-site.

This rig and its contents were intended to produce a 'flashed-over' fire in a relatively short time (in the event the whole experiment, from ignition to manual extinguishment lasted around one hour). A type of *Event* in terms of the FireGrid ontology, flashover typically occurs when the gases produced by a fire in some enclosed space reach temperatures high enough (above 500°C,as a rule of thumb) to ignite simultaneously all combustible matter in the vicinity. From the perspective of responders, flashover represents a potential transition from a contained fire to an uncontrolled fire. In addition, certain structural elements of the rig were expected to deform and fail during the fire; the potential collapse of ceilings and floors is, of course, a major hazard for fire-fighters.

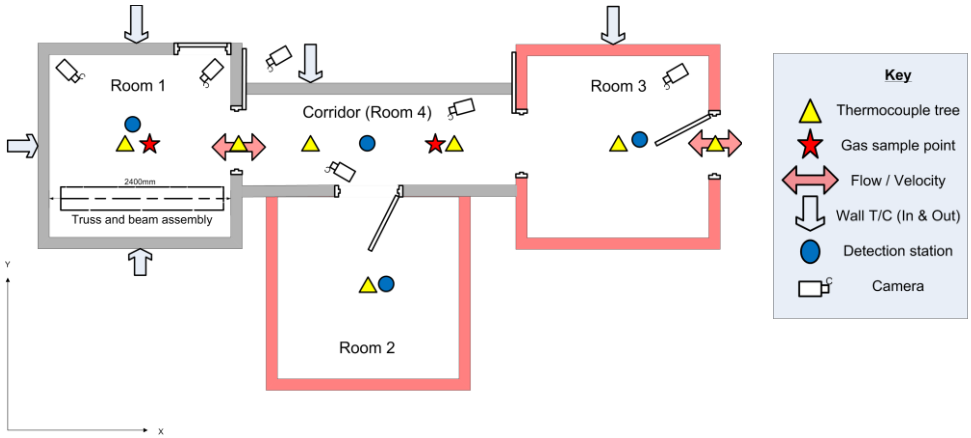


**Fig. 3 The experimental rig (based on an original diagram drawn by P. Clark, BRE).**

## 5.2 Experimental Results

Figs. 4-7 show camera footage of the experiment along with screenshots of the user interface, and figs. 8-11 show some of the underlying results from the principal simulation model. During the experiment the computational architecture functioned entirely as envisaged and the K-CRISP model was able to continuously generate forecasts of the future evolution of the hazard (so as not to overwhelm the end user with varying predictions, it was chosen to update the user interface display at an interval of 15s).

Fig. 4 is a four-camera view of the fire scene after fire has been detected. Fig. 5 presents interpretation of the incident at the same time as seen on the IC's C3I interface. The fire in Room 1 has been detected by the system (indicated by the red floor in this room), and the models have been invoked. While the current conditions throughout the rig may still be considered tenable for fire-fighting operations (the lower 'traffic light' is coloured amber in the fire room and green elsewhere), information from the model indicates that the conditions in the Room 1 are predicted to deteriorate to 'red' (and those in the Corridor to 'amber'). The pop-up window here shows details of the various red and amber hazards predicted for Room 1 over the next 15 minutes. As it happened, there then followed a very interesting course of events, with the fire reaching a size, at about 15 minutes after detection, where it was poised to flash-over, but due to the fuel burning out, there then in fact followed a decay phase. During the decay the model adapted to the change in burning behaviour by revising its predictions accordingly, retracting the prediction of flashover. However, nearly 20 minutes later, another sudden transition in the burning behaviour occurred, with fire spreading to further items (table and TV unit), with the resulting increase in size then supporting rapid further growth, igniting the bookshelf and leading to flashover. Fig. 6 shows the fire scene after flashover and Fig. 7 shows the C3I presentation of the same situation. The latter suggests that current and projected conditions in every room are now 'red', with high temperatures recorded throughout the rig – though it should be noted that due to the speed of this transition there was no advanced warning of the precise moment of the transition. Indeed, to truly predict such would require a model of enormously increased detail, in order to represent the ignition of the further items, a capability that would be impractical with all the uncertainties of real-life scenarios [17]. However, the essence of the demonstration concerns the general capabilities of the model to provide an indication of the possible evolution of the incident, rather than detailed information about the precise moment of the transition in burning behaviours, which would in any case be far too late for reaction or evasive action by the fire service. In this respect, the model was successful in having given an early warning of the possible evolution of the fire early on in the incident, and even if the fire had proceeded to burn out completely before the flashover transition had been reached that warning would have been no less valid.

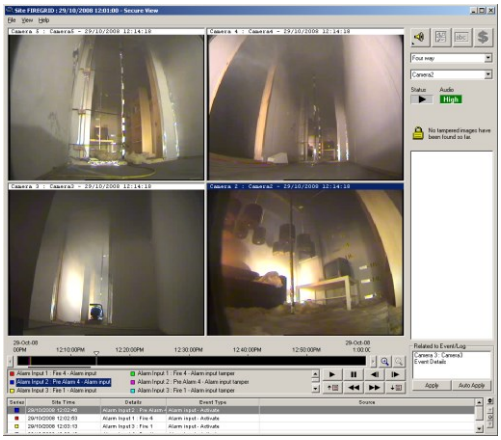

**Fig. 4 Camera footage from within the apartment after the fire has taken hold.**

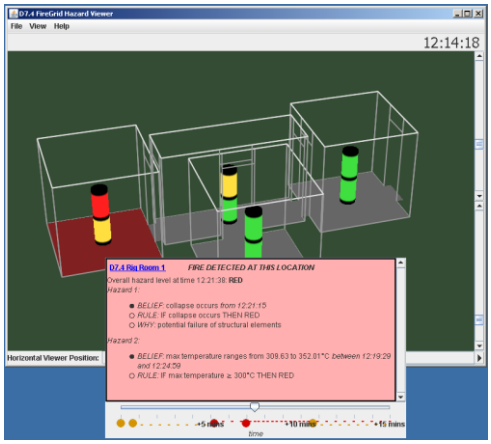


**Fig. 5 The C3I interface interpreting for the IC the state of the incident as it is shown in Fig. 4.**

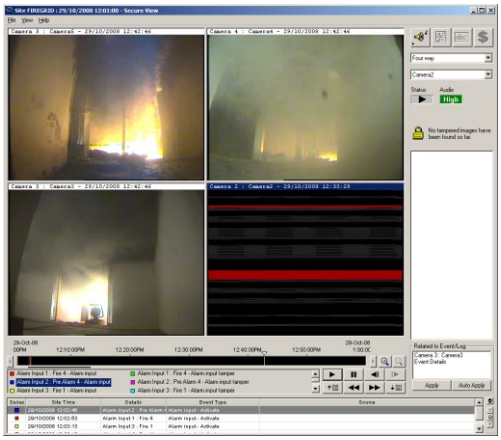

**Fig. 6 Camera footage of the incident immediately after flashover has occurred – flame, smoke and debris fill the apartment (and one camera has been destroyed).**

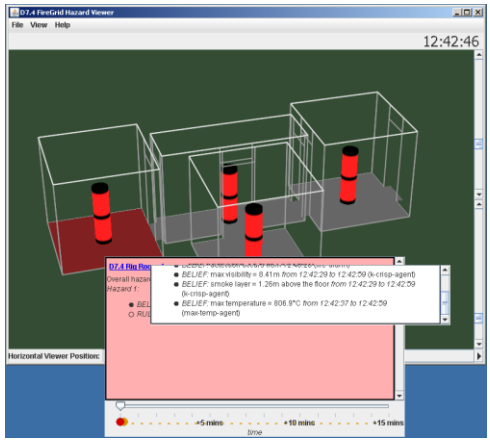


**Fig. 7 The C3I interface showing the interpretation of the incident state at the same time as the footage shown in Fig. 6.**

Subsequent to the live tests, further analysis of model performance was undertaken using a replay of the sensor data generated during the experiment, that is, with the model is still effectively 'blind', and with equivalent computational resources. Figs. 8 – 11 show the comparisons between sensor measurements and prediction results at selected times during the evolution of the incident. The dark line is the average of the actual temperature values recorded in Room 1 up to the time in question; the light grey lines show the projected values of the corresponding parameter in each of an ensemble of cases selected by the model at any particular instant, drawn from a much greater number of scenarios generated (new scenarios were computed at a rate of about 1000 per minute using only 8 nodes on the HPC resource). It can be seen that selection from amongst the range of current scenarios is adequate to allow the model to follow the general trend of the fire evolution, even when the course of the fire changed completely as between Figs. 9 and 10. Tests also demonstrated that the model was very robust to sensor failure, continuing to perform adequately with even when live sensor inputs had been reduced to a single characteristic hot layer gas temperature in the room of fire origin.

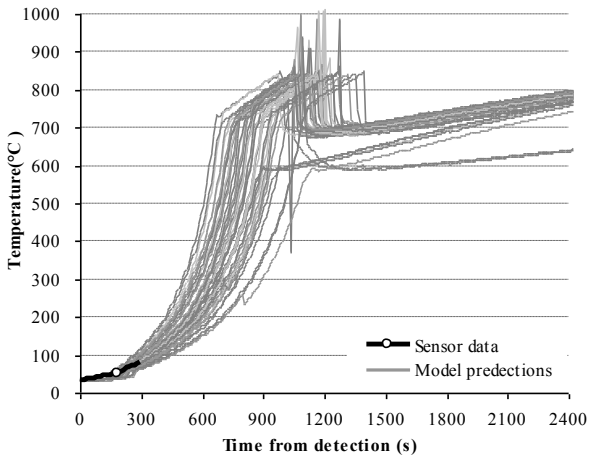


**Fig. 8 Comparison between actual sensor measurement and prediction result at 300s after detection.**

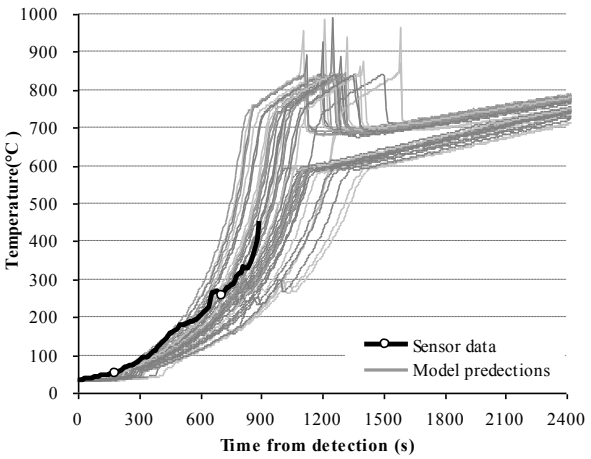


**Fig. 9 Comparison between actual sensor measurement and prediction result at 900s.**

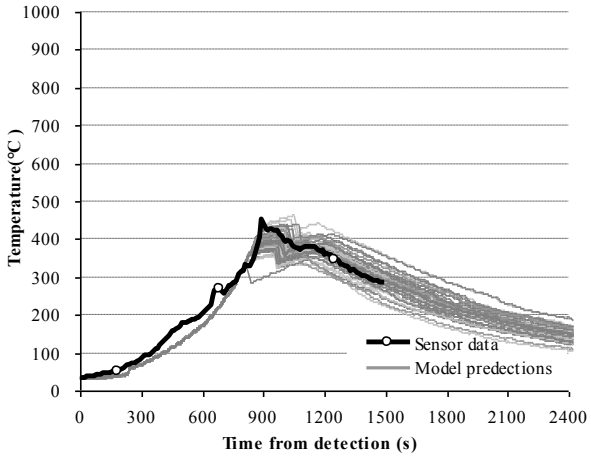


**Fig. 10 Comparison between actual sensor measurement and prediction result at 1500s.**

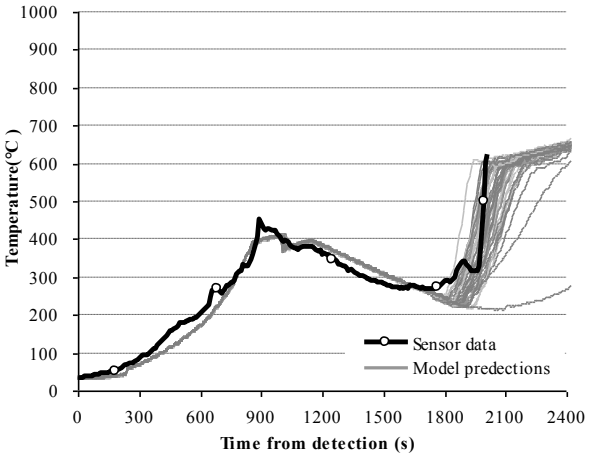


**Fig. 11 Comparison between actual sensor measurement and prediction result at 2010s.**

Besides the fire status information the model was also generating predictions of structural integrity. During the live test these correctly indicated probable failure of a structural truss placed within the room of fire origin, as actually occurred towards the end of the fire. Again, more careful analysis was undertaken later using a replay of the sensor data. Figs. 12 and 13 compare the evolution of the experimental measurement and predictions of the vertical deflection at the centre of the truss, with the predictions being a weighted average derived from the range of fire curves generated at any particular instant. For this member the transition from collapse being possible to being probable occurs at a mid-span deflection of 0.25mm. Fig. 12 shows the prediction early in the fire at a time of 300s after detection. Even at this stage the greatly increased likelihood of collapse at times approaching 900s is indicated, with the transition from possible to probable occurring at around 870s; also shown is the actual measured deflection – this is initially slightly negative (that is, upward) presumably due to the fire exposure of the lower surfaces of the member, a factor not considered in the simple model; nevertheless, the actual behaviour was

comparable to the prediction, with a deflection of 0.25mm reached at 814s. The failure values are also in accordance with the fire temperature rising toward 500°C, beyond which steel begins to rapidly lose strength.

After the fire behaviour changed at around 900s, the predicted structural performance was revised (Fig. 13), with some recovery of the member up to a time of about 1900s, a trend also mirrored in the deflection measurements; however, following flashover at around 2000s a rapid further deterioration ensued, induced by the sudden increase in fire exposures. Total collapse of the truss followed at around 3100s during the cooling phase of the fire, extinguishing water having been applied at about 2500s.

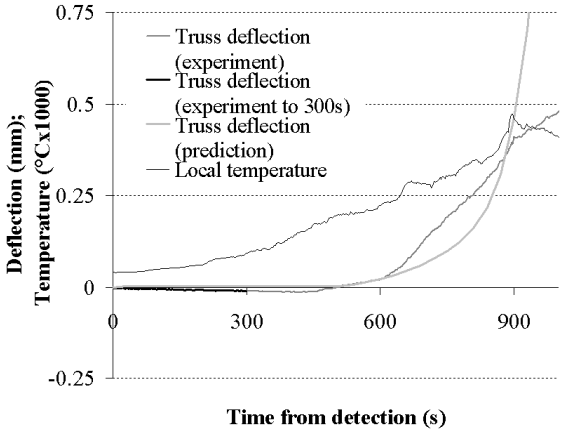


**Fig. 12 Comparison between actual sensor measurement and predicted structural deflections at 300s after detection.**

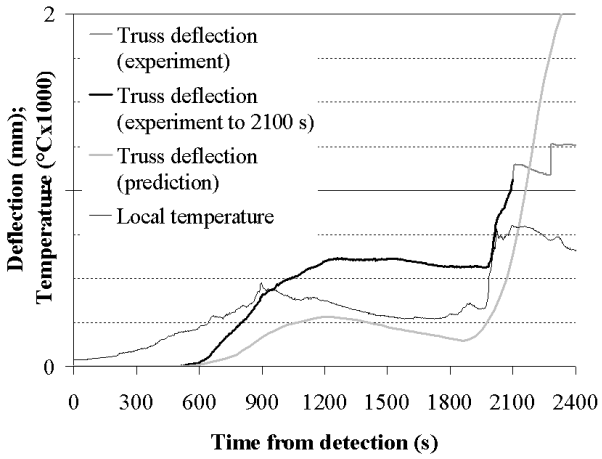


**Fig. 13 Comparison between actual sensor measurement and predicted structural deflections at 2100s.**

## 5.3 Summary

The experiment was sufficient to demonstrate that our proposed architecture is capable of providing real-time decision-making information and it therefore fulfilled its primary objectives. The various technical components were integrated seamlessly, which can be described from the following aspects:

- The latest sensor measurements were relayed rapidly via simple interpretative models to the end user, to inform of any current hazards.
- The sensor data was consumed and assimilated by the K-CRISP predictive simulation model. The model adapted to the evolving sensor measurements and predicted hazard conditions, encompassing both fire exposures and structural integrity, which were fully consistent with the range of possible outcomes exhibited by the actual fire.
- Access to the HPC resources through the Grid proceeded without any delay, which ensured that prediction results were generated and delivered in good time (that is, while they were still considerable 'actionable' predictions).

- The C3I has provided a succinct interface for Incident Commanders using a ‘traffic light’ display to convey hazard information in an accessible manner, supplemented with underlying rationale and advice, and designed to integrate into the operational decision-making processes.

## 6. Conclusions

Emergency response is an important concern in modern societies; it requires the urgent and coordinated invocation of resources including human responders, organisations and relevant services in a timely and effective manner. Technologies like those mentioned in this paper could come to play an effective role in any response – and, indeed, they have the potential to transform the way in which responders approach their task.

Through an imaginative integration of advanced sensors, computational models, Grid and multi-agent system technologies, we proposes a novel e-Infrastructure for next-generation emergency response support: live sensor data captures the unfolding situation; the Grid enables uniform and secure access to distributed HPC resources; simulation models running on these resources produce interpretations and predictions which are further interpreted by knowledge-based reasoning to relate them to current and projected hazards; and an agent layer seamlessly delivers these interpretations to the users of the system as decision-making intelligence. The experimental results on a large-scale fire suggest that the FireGrid approach has the potential to provide vital information to emergency responders to support critical decision making and contribute to effective emergency management.

## Acknowledgments

The work reported in this paper has formed part of the FireGrid project. This project is co-funded by the UK Technology Strategy Board’s Collaborative Research and Development programme, following an open competition. The authors acknowledge the contribution of all colleagues in the FireGrid project. This work made use of the facilities of HPCx, the UK’s national high-performance computing service, which is provided by EPCC at the University of Edinburgh and by STFC Daresbury Laboratory, and funded by the Department for Innovation, Universities and Skills through EPSRC’s High End Computing Programme. The University of Edinburgh, project partners and project funding agencies are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

Figs. 4 and 6 are screenshots of the SecureView tool, which is copyright of Xtralis Pty. Ltd.

The authors would like to thank the anonymous reviewers, who provided detailed and constructive comments on an earlier version of this paper.

## References

[1] C.E. Alchourròn, P. Gärdenfors, D. Makinson, On the Logic of Theory Change: Partial Meet Contraction and Revision Functions, Journal of Symbolic Logic, 50 (1985) 510-530.

[2] ANSYS CFX, http://www.ansys.com/products/default.asp, last accessed: 8 June 2010.

[3] D. Berry, A. Usmani, J. Torero, A. Tate, S. McLaughlin, S. Potter, A. Trew, R. Baxter, M. Bull, M. Atkinson, FireGrid: Integrated Emergency Response and Fire Safety Engineering for the Future Built Environment, UK e-Science Programme All Hands Meeting (AHM-2005), Nottingham, UK, 2005, http://www.allhands.org.uk/2005/proceedings/papers/384.pdf, last accessed: 8 June 2010.

[4] U. Cortes, M. Sanchez-Marre, L. Ceccaroni, Artificial intelligence and environmental decision support systems. Applied Intelligence, 13(1) (2000) 77-91.

[5] FireGrid, http://www.firegrid.org/, last accessed: 4 June 2010.

[6] I. Foster, Globus Toolkit Version 4: Software for Service-Oriented Systems, IFIP Int. Conf. Network and Parallel Computing, in: H. Jin, D.A. Reed, W. Jiang (Eds.), LNCS 3779, Springer-Verlag, 2005, pp. 2-13.

[7] I. Foster, C. Kesselman, The Grid: blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999.

[8] J. Fraser-Mitchell, An object-oriented simulation (crisp II) for fire risk assessment, Fire Safety Science, 4 (1994) 793-804.

[9] J. Fraser-Mitchell, Risk assessment of factors relating to fire protection in dwellings. Fire Safety Science, 5 (1997) 631-642.

[10] S. Fuhrmann et al., Collaborative emergency management with multimodal GIS, ESRI International User Conference, San Diego, CA. 2003, http://www.geovista.psu.edu/publications/2003/Fuhrmann_ESRI_03.pdf, last accessed: 8 June 2010.

[11] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, Sweetening Ontologies with DOLCE, in: A. Gómez-Pérez, V.R. Benjamins (Eds.), Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Springer Verlag, LCNS 2473, 2002, pp. 166-181.

[12] A.V. Gheorghe, D.V. Vamanu, Adapting to new challenges: IIDS for emergency preparedness and management, Int. J. Risk Assessment and Management, 2 (2001) 211-223.

[13] J.Z. Hernandez, J.M. Serrano, Knowledge-based models for emergency management systems, Expert Systems with Applications, 20 (2001) 173-186.

[14] HM Fire Service Inspectorate, Fire Service Manual, Volume 2: Fire Service Operations, Incident Command, HM Fire Services Inspectorate Publications, The Stationary Office, London, 2002.

[15] G. Jing, Ragbus: A universal decision support platform based on intelligent reflective agents, in: Proc. Biennial Meeting of the Environmental Modelling and Software (iENSs2002), Lugano, Switzerland, 2002, pp. 3:390-395.

[16] S.-H. Koo, J. Fraser-Mitchell, R. Upadhyay, S. Welch, Sensor-linked fire simulation using a Monte-Carlo approach, Proc. 9th Int. Symp. Fire Safety Science, Karlsruhe, Germany, 2008.

[17] S.-H. Koo, J. Fraser-Mitchell, S. Welch, Sensor-steered fire simulation, Fire Safety Journal, 45 (2010) 193-205, http://dx.doi.org/10.1016/j.firesaf.2010.02.003

[18] G. von Laszewski, I. Foster, J. Gawor, P.A. Lane, Java Commodity Grid Kit, Concurrency and Computation: Practice and Experience, 13 (2001) 643-662.

[19] D. Massaguer, V. Balasubramanian, S. Mehrotra, N. Venkatasubramanian, Multiagent simulation of disaster response, Proc. 5th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'06), Hakodate, Japan, 2006.

[20] S. Potter, A. Tate, G. Wickler, Using I-X panels as intelligent to-do lists for agent coordination in emergency response, Proc. 3rd Int Conf. Information Systems for Crisis Response and Management (ISCRAM 2006), Newark, NJ, USA, 2006, pp. 272-281.

[21] S. Potter, G. Wickler, Model-based query systems for emergency response, in: F. Fiedrich, B. Van de Walle (Eds.), Proc. 5th Int. Conf. Information Systems for Crisis Response and Management (ISCRAM 2008), Washington DC, USA, 2008.

[22] T. Schoenharl, G. Madey, WIPER: A multi-agent system for emergency response, in: B. Van de Walle, M. Turoff (Eds.), Proc. 3rd Int. Conf. Information Systems for Crisis Response and Management (ISCRAM 2006), Newark, NJ, USA, 2006.

[23] J.M. Schurr, N. Kasinadhuni, M.Tambe, J.P. Lewis, P. Scerri, The defacto system for human omnipresence to coordinate agent teams: The future of disaster response, in: Proc. 4th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'05), Utrecht, Netherlands, 2005.

[24] A. Stone, Powerful combination: GIS and web services. IEEE Distributed Systems Online, 5(2) (2004) 1541-1545.

[25] N.A. Theophilopoulos, S.G. Efstathiadis, Y. Petropoulos, Envisys environmental monitoring warning and emergency management system. Spill Science and Technology Bulletin, 3(1/2) (1996) 19-24.

[26] S. Tufekci, An integrated emergency management decision support system for hurricane emergencies, Safety Science, 20 (1995) 39-48.

[27] G. Wickler, S. Potter, Information-gathering: from sensor data to decision support in three simple steps, Intelligent Decision Technologies, 3 (2009) 3-17.